

REMARKS

Status of Claims:

In this application, claims 1-59 are currently pending. Claims 1, 5, 23, 37, and 50 are amended by this Response. In some instances, these claims have been amended to more particularly point out the distinctions between the present invention and the prior art. In other cases, the claims have been amended to remove unnecessarily restrictive claim limitations. Claims 2-4, 6-22, 24-36, 38-49, and 51-56 have not be altered since filing. Claims 57-59 have been added. No claim has been deleted. Entry of these amendments is respectfully requested.

Claim Rejections Under 35 U.S.C. 102

Overview of Anderson, "Serverless Network File Systems"

The current office action has rejected claims 1-56 as being anticipated by T. Anderson, et al., "Serverless Network File Systems," Proceedings of the Fifteenth ACM Symposium on Operating System Principals, 1995 ("Anderson"). The Anderson reference teaches a serverless network file system known as xFS. The xFS system was acknowledged as a prior art file system on page 5, line 14 of the Specification for the present application. Anderson describes multiple roles that a computer can perform in the xFS system, namely clients, managers, cleaners, or storage servers. Anderson 3.1. In fact, an individual computer can perform one, two, three, or all four of these roles at the same time. Id.

The applicant understands that the clients of Anderson could be considered similar the client computers of the present claims, in that they handle file service requests received from applications. Similarly, the managers of Anderson could be considered similar to the server computer of the present claims in that they handle requests from a variety of clients relating to the particular files that they manage. Nonetheless, important distinctions exist between the Anderson file system and the current claims. In particular, the claims define at various points a file system where the server software on the server computer manages the namespace (in part by receiving namespace requests made by the clients), while the clients directly retrieve, analyze, and/or modify file system metadata. A review of the Anderson reference reveals that the

xFS file system described by Anderson completely reverses these roles. In effect, both Anderson and the pending claims are similar in that they describe a file system with clients and servers each performing different functions, but they are fundamentally opposite in how those functions are assigned. This can be seen in the following table, which summarizes the differences in various file systems with respect to metadata management and namespace management. In this table, “client-based” means that the computer running the application is the computer that performs the namespace and/or metadata management operations.

	Server-Based Namespace	Client-Based Namespace
Server-Based Metadata	(other server-based shared storage file systems)	Anderson xFS
Client-Based Metadata	The present invention	(other serverless shared storage file systems)

In Anderson, the clients do not analyze the metadata, are not allowed to modify the metadata, and in fact are not even given access to the metadata. The xFS metadata described by Anderson (including the data block locations of the read data) is stored in Index Nodes (commonly referred to as inodes). Anderson, 3.1.2. Only the managers are allowed to access and manipulate these Index Nodes in Anderson. Anderson, 3.2.1.¹

While the managers are responsible for file metadata in the Index Nodes, it is the clients that control the namespace in Anderson. A file can be retrieved from storage only once the file’s index number is determined. Given the file’s index number, the managers can request that the appropriate data be sent to the requesting clients. Anderson, 3.2.1 and Figure 3. It is the clients in Anderson that are given responsibility for converting the namespace designation of the file into the appropriate index number. This is accomplished by reading a file’s parent directory. Anderson, 3.1.3. The directory is stored as a file alongside the regular files in file system. Id. To find the parent directory file, the grandparent directory file must first be read, with the recursion continuing all the way to the root.

¹ See the last paragraph of 3.2.1, where Anderson states that “[o]ne important design decision was to cache index nodes at managers, not at clients” and “only the manager handling an index number directly accesses its index node”

Anderson, 3.2.1 (second paragraph). The method used by the client to obtain a file is shown in Figure 3 and described at section 3.2.1. The manager itself treats all file requests identically, and therefore does not know whether the client is requested a regular file accessed by a user application or a directory file. Id. In essence, the manager is completely oblivious to the namespace as it deals only with index numbers--only the client is capable of converting file names in the namespace into index numbers that can be used by the manager to file the real data. Anderson, 3.2.1 and Figure 3.

In the present office action, the Examiner has specifically referred to configurations where all computers run both client and manager roles. Assigning multiple roles to each computer, however, does not make that the interaction between computers the same as the present invention. A combined client and manager computer in Anderson will be responsible for i) interfacing with the application program; ii) managing the namespace for local requests (through the client); and iii) managing metadata for local and remote requests (through the manager). However, this combined client and manager computer will never manage the namespace for remote requests (i.e., requests received from another computer), nor will it ever request another computer (i.e., a remote server) to manage the namespace for local requests.

Serverless File Systems and Name Space Inefficiencies

The fact that Anderson relies on clients to control and manipulate the namespace in the xFS file system is consistent with the approach taken by other serverless file systems. As the Applicant has explained in previous communications with the PTO in the parent of the present application (now issued as U.S. Patent No. 6,697,846), prior art shared storage file systems advanced over the years from server-based file systems that rely upon central file manager computers to serverless file systems where client computers perform local, file management tasks. In prior art *server-based* designs, file manager computers managed namespace and metadata. These server-based designs incurred several problems, including lack of scalability of metadata operations as additional clients were added to the environment, slow performance as client requests were queued for server response, and single points of failure. As time

moved on, *serverless* designs were developed in which client computers performed metadata generation and modifications for all file accesses of that client, and directly handled namespace data.

The main disadvantage of prior art serverless designs relates to the inefficient namespace management incurred when multiple clients cooperate to maintain directory files.² In these systems, directory lookup times are directly proportional to directory tree depths for non-cached directory files. In a multi-computer environment, maintaining coherency of cached directories is a significant performance obstacle. In other words, a directory lookup that extends through ten layers of directories will take approximately ten times longer than a directory lookup at the root directory. This is a significant performance issue for serverless shared storage file systems because of the ubiquitous nature of namespace operations during the operation of a file system. In addition, these types of client-based directory lookups are a significant source of multiple client contention. Given that nearly every file operation requires at least one namespace operation, these performance and contention issues adversely influence the performance of the whole file system when namespace control is distributed across multiple clients.

The present invention avoids the namespace problem inherent in serverless shared storage file systems through the use of the meta-data file server that handles the namespace for the file system. At the same time, the present invention avoids the most significant problems with server-based shared storage file systems by having clients directly generate and alter meta-data. This solution is not obvious. The approach taken by Anderson is exactly the opposite of that set forth in the pending claims. Anderson requires a separate file read for each directory layer because the clients in Anderson handle the namespace by separately reading each directory as a separate file. Furthermore, Anderson fails to take advantage of the benefits of direct client access to metadata (which is

² It is not clear that this disadvantage of serverless design was known in the prior art before the present invention. The inventor of the present invention was personally involved with the development of the GFS serverless file system. Only after three years of research did it become clear to the inventor that serverless designs experience poor namespace cache efficiencies.

accomplished by other serverless file systems), since Anderson keeps all inode (Index Node) access, interpretation, and alteration at the manager level.

Analysis of the Pending Claims

In distinguishing the pending claims from Anderson, the Applicant has remembered the Examiner's reference to configurations in Anderson where all computers run both client and manager roles. Even in such configurations, Anderson does not teach or suggest numerous aspects of the pending claims.

Independent Claims 1, 9, 23, 43, 52, and 54: namespace requests passing from client to server. These independent claims recite "namespace requests" or "directory requests" moving from a client to a server computer. These requests (such as requests to add new filenames to the namespace, to remove existing filenames from the namespace, to list the contents of directory files, or to search the namespace for filenames) move between computers in these pending claims. This claim limitation is not found in Anderson. Even where a single computer operates as a client and a manager, no computer in Anderson sends namespace requests to a separate computer. The clients in Anderson handle the namespace, and since every node other than storage server nodes has a client (Anderson, caption to Figure 2), no namespace request would ever leave that computer. These claims are distinguished from typical server-based systems by having the client (as opposed to the server) directly access, analyze, or modify meta-data for the file system. This latter characteristic also distinguishes the clients defined in these claims from the clients in Anderson.

Independent Claims 31 and 50: client's use of an indirect pointer received from a server. Claim 31 defines a particular way in which a client uses an indirect pointer received from a server computer "to obtain the file related direct pointer directly from the storage device," and then uses the file related direct pointer to read and write real-data directly from the storage device. Similarly, claim 50 defines a method in which a client computer requests an indirect extent pointer from a server computer, uses the indirect extent pointer to retrieve metadata from the storage device, and then analyzes the metadata to determine real-data locations. Claim 50 further requires that the client is able to allocate additional storage space for the file and update metadata on the storage

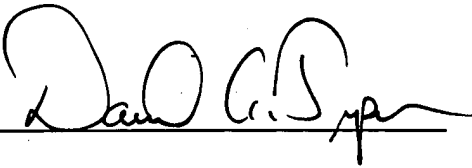
device for a file read request. This handling of file location metadata is directly opposite to that described in Anderson, where the managers handle all data pointers in the index nodes and the clients merely request data based on an index number and then received the correct data from the storage devices. (Anderson, Figure 3). This is true even when Anderson is configured to have computers with both clients and managers, since no indirect pointers or indirect extent pointers are ever passed between computers in Anderson regardless of configuration. No other prior art systems store indirect pointers maintained by a server and used by clients to access real data "within the namespace" maintained by the server (as required by claim 31). Furthermore, no other prior art file system use indirect extent pointers that are forwarded from the server to the client to allow the client to directly access, modify, and store metadata on the storage device (as required by claim 50).

Independent Claim 37: client handling allocation and de-allocation. In claim 37, a server computer maintains a namespace, and a client computer that receives requests from a local application program is responsible for handling allocation and de-allocation of regular files in the file system. The office action broadly cites Figure 2, and sections 1, 3.1, 3.1.1, and 3.2.2 for this teaching in Anderson. The Applicant has reviewed these portions of the Anderson reference and does not find these teachings or any related teaching of the handling of allocation and de-allocation in Anderson.

Dependent Claims 58-59: shared storage device. These claims require the presence of a shared storage device where the client computers directly share data on the shared storage device. This is not found in Anderson, as Anderson uses a storage server in which a server interposes itself between the storage device and the network. See, e.g., Anderson 3.1.4.

CONCLUSION

All of the claims remaining in this application should now be seen to be in condition for allowance. The prompt issuance of a notice to that effect is solicited.

Date: 

Respectfully Submitted,

Dec - 10, 2008
Daniel A. Tysver
Registration No. 35,726
Beck & Tysver, P.L.L.C.
2900 Thomas Avenue South, #100
Minneapolis, MN 55416
Telephone: (612) 915-9634
Fax: (612) 915-9637